
Course Code	**:**	**MCS-051**
Course Title	**:**	**Advanced Internet Technologies**
Assignment Number	**:**	**MCA (5)/051/Assign/2013**
Maximum Marks	**:**	**100**
Weightage	**:**	**25%**
Last Dates for Submission	**:**	**15th October, 2013 (For July 2013 Session)** **15th April, 2014 (For January 2014 Session)**

There are eight questions in this assignment. Each question carries 10 marks. Rest 20 marks are for viva-voce. Answer all the questions. You may use illustrations and diagrams to enhance the explanations. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation.

Question1:

Assume that there is a table named as product in oracle with (10 marks) the following fields (Prod-ID, Product-name, Price, Vender-name) Write a Java Programme to insert and then display the records of this table using JDBC.

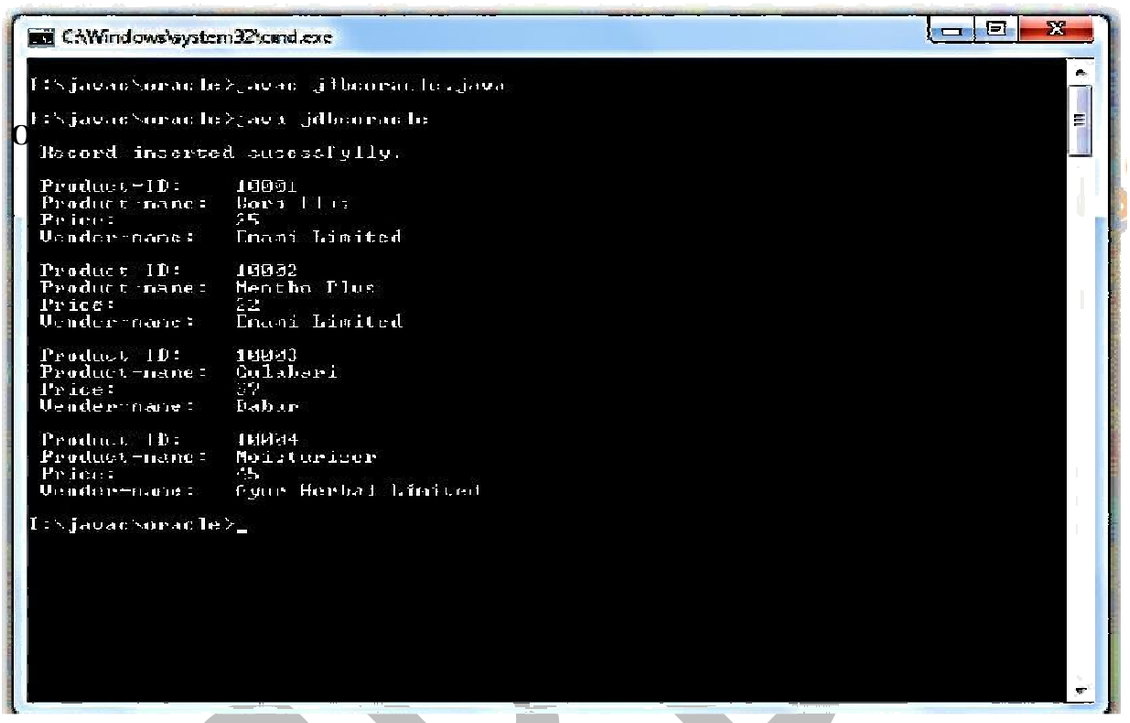
Answer:

```
import java.sql.*;

public class jdbcoracle {

        public static void main(String[] args) {

        try {
                Class.forName("oracle.jdbc.driver.OracleDriver");

                } catch (ClassNotFoundException e)
                        { e.printStackTrace();
                        return;
                }

                Connection connection = null;

                try {

                connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","lokesh");

                        try {

                        String insertTableSQL ="INSERT INTO PRODUCT Values(?,?,?,?)";
```

```
                    PreparedStatement preparedStatement = null;

                    preparedStatement = connection.prepareStatement(insertTableSQL);
                    preparedStatement.setInt(1,10004);
```

```
                        preparedStatement.setString(2,"Moisturizer");
                        preparedStatement.setInt(3,45);
                        preparedStatement.setString(4,"Ayur Herbal Limited");

                        preparedStatement.executeUpdate();
                        System.out.println("\n Record inserted
                        sucessfylly."); preparedStatement.close();
                        }

                        catch (SQLException s)
                        {
                                System.out.println("\n Record not inserted sucessfylly.");
                        }

                                try {
                                Statement st = connection.createStatement();
                                ResultSet res = st.executeQuery("SELECT * FROM product");

                                while (res.next()) {
                                        int i = res.getInt("Product-ID");
                                        String n = res.getString("Product-
                                        name"); int p = res.getInt("Price");
                                        String v = res.getString("Vender-name");

                                System.out.println("\n Product-ID:\t" + i + "\n Product-name:\t"
+ n + "\n Price:\t\t" + p + "\n Vender-name:\t" + v);
                                }
                                connection.close();
                                st.close();
                                res.close();
                                }
                                catch (SQLException s){
                                System.out.println("\n SQL code does not execute.");
                                }

                } catch (SQLException e) {
                        e.printStackTrace();
                        return;
                }

                if (connection == null) {
                        System.out.println("\n Failed to fetch records from oracle database");
                }
        }
```

}



Question 2:

(a) Write an XML DTD to represent the Grade Card of a student which contains: **(5 marks)**
(i) Name- Last, Middle, and First
(ii) Subjects- Five subjects
(iii) Assignments marks
(iv) Total Marks
(v) Result- Pass/Fail

Answer:

XML Code for Grade Card:

<?xml version="1.0"?>

<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

```
<!-- grade.xml -->
<!-- Representing the Grade Card of student in XML document -->
<!DOCTYPE grade SYSTEM
grade.dtd"> <student>
        <grade enroll="011223344">
<name>
        <lname>SINGH</lname>
        <mname>CHANDRA</mname>
        <fname>LOKESH</fname>
</name> <subject>
        <sub_a>MCS011</sub_a>
        <sub_b>MCS012</sub_b>
        <sub_c>MCS013</sub_c>
        <sub_d>MCS014</sub_d>
        <sub_e>MCS015</sub_e>
</subject>
<sub_marks>
        <marks_a>90</marks_a>
        <marks_b>91</marks_b>
        <marks_c>92</marks_c>
        <marks_d>93</marks_d>
        <marks_e>94</marks_e>
</sub_marks>
<assinment_marks>
        <assign_a>95</assign_a>
        <assign_b>96</assign_b>
        <assign_c>97</assign_c>
        <assign_d>98</assign_d>
        <assign_e>99</assign_e>
</assignment_marks>
<total_marks>446</total_marks>
<result>Pass</result>
</grade>
<content>This is system generated Grade Card has not any leagle
value</content> </student>
```

DTD Code for XML Grade Card:

```
<?xml version="1.0" rmd="internal"?>
</ELEMENT student(grade +,content *)>
<!Element grade(name, subject, sub_marks, assignment_marks, total, result)>

<!ATTLIST grade enroll CDATA #IMPLIED>
```

<!Element name(lname, mname, lname)>
<!Element lname(#PCDATA)>
<!Element mname(#PCDATA)>
<!Element fname(#PCDATA)>

<!Element subject(sub_a, sub_b, sub_c, sub_d,
sub_e)> <!Element sub_a(#PCDATA)>
<!Element sub_b(#PCDATA)>
<!Element sub_c(#PCDATA)>
<!Element sub_d(#PCDATA)>
<!Element sub_e(#PCDATA)>

<!Element sub_marks(marks_a, marks_b, marks_c, marks_d,
marks_e)> <!Element marks_a(#PCDATA)>
<!Element marks_b(#PCDATA)>
<!Element marks_c(#PCDATA)>
<!Element marks_d(#PCDATA)>
<!Element makrs_e(#PCDATA)>

<!Element assignment_marks(assign_a, assign_b, assign_c, assign_d,
assign_e)> <!Element assign_a(#PCDATA)>
<!Element assign_b(#PCDATA)>
<!Element assign_c(#PCDATA)>
<!Element assign_d(#PCDATA)>
<!Element assign_e(#PCDATA)>

<!Element total_marks(#PCDATA)>
<!Element result(#PCDATA)>
<!Element content(#PCDATA)>

(b) How does Session bean different from Entity bean in terms of object sharing and failure recovery? **(5 marks)**

Answer:

Different from Entity bean in terms of object sharing and failure recovery:

Functional Area	**Session Bean**
Object state	Maintained by the container in the main memory across transactions. Swapped to secondary storage when deactivated.
Object sharing	A session object can be used by only one client.
State externalisation	The container internally

maintains the

	session object's state. The state is inaccessible to other programs.	**Entity Bean** Maintained in the database or other resource manager. Typically cached in the memory in a transaction.
Transactions	The state of a session object	An entity object can be shared by multiple clients. A client may pass an object reference to another client.
		The entity object's state is typically stored in a database. Other programs, such as an SQL query, can access the state in the database.
		The state of an entity object i

---

	can be synchronised with a transaction but is not recoverable.	typically changed transactionally and is recoverable.
Failure recovery	A session object is not guaranteed to survive failure and restart of its container. The references to session objects held by a client becomes invalid after the failure.	An entity object survives the failure and the restart of its container. A client can continue using the references to the entity objects after the container restarts.

Question 3:

Explain four basic mechanisms through which a web client can authenticate a user to a web server during HTTP authentications. **(10 marks)**

Answer:
A web client can authenticate a user to a web server using one of the following mechanisms:

a) HTTP Basic Authentication b) HTTP Digest Authentication
c) Form Based Authentication
d) HTTPS Client Authentication

a) HTTP Basic Authentication:

HTTP Basic Authentication, which is based on a username and password, is the authentication mechanism defined in the HTTP/1.0 specification. A web server requests a web client to authenticate the user. As a part of the request, the web server passes the realm (a string) in which the user is to be authenticated. The realm string of Basic Authentication does not have to reflect any particular security policy domain (confusingly also referred to as a realm). The web client obtains the username and the password from the user and transmits them to the web server. The web server then authenticates the user in the specified realm. Basic Authentication is not a secure authentication as user passwords are sent in simple base64 ENCODING (not ENCRYPTED !), and there is no provision for target server authentication. Additional protection mechanism can be applied to mitigate these concerns: a secure transport mechanism (HTTPS), or security at the network level (such as the IPSEC protocol or VPN strategies) can be deployed.

b) HTTP Digest Authentication:

Similar to HTTP Basic Authentication, HTTP Digest Authentication authenticates a user based on a username and a password. However, the authentication is performed by transmitting the password in an ENCRYPTED form, which is much MORE SECURE than the simple base64

encoding used by Basic Authentication, e.g., HTTPS Client Authentication. As Digest Authentication is not currently in widespread use, servlet containers are encouraged but NOT REQUIRED to support it.

---

c) Form Based Authentication:

The look and feel of the login screen' cannot be varied using the web browser's built in authentication mechanisms. This form based authentication mechanism allows a developer to CONTROL the look and feel of the login screens. The web application deployment descriptor, contains entries for a login form and error page. The login form must contain fields for entering a username and a password. These fields must be named j_username and j_password, respectively.

When a user attempts to access a protected web resource, the container checks the user's authentication. If the user is authenticated and possesses authority to access the resource, the requested web resource is activated and a reference to it is returned. If the user is not authenticated, all of the following steps occur:

1) The login form associated with the security constraint is sent to the client and the URL path triggering the authentication stored by the container.
2) The user is asked to fill out the form, including the username and password fields.
3) The client posts the form back to the server.
4) The container attempts to authenticate the user using the information from the form.
5) If authentication fails, the error page is returned using either a forward or a redirect, and the status code of the response is set to 200.
6) If authentication succeeds, the authenticated user's principal is checked to see if it is in an authorised role for accessing the resource.
7) If the user is authorised, the client is redirected to the resource using the stored URL Path.

The error page sent to a user that is not authenticated contains information about the failure. Form Based Authentication has the same lack of security as Basic Authentication since the user password is transmitted as a plain text and the target server is not. Authenticated. again additional protection can alleviate some of these concerns: a secure transport mechanism (HTTPS), or security at the network level (such as the IPSEC protocol or VPN strategies) are applied in some deployment scenarios.

Form based login and URL based session tracking can be problematic to implement. Form based login should be used only when, sessions are being maintained by cookies Or by SSL session information.

d) HTTPS Client Authentication:

End user authentication using HTTPS (HTTP over SSL) is a strong authentication mechanism. This mechanism requires the user to possess a Public Key Certificate (PKC). Currently, PKCs are useful in e-commerce applications and also for a single sign-on from within the browser. Servlet containers that are not J2EE technology compliant are not required to support the HTTPS protocol. Client-certificate authentication is a more secure method of authentication than either BASIC or FORM authentication. It uses HTTP over SSL, in which the server and, optionally, the client authenticate one another with Public Key Certificates. Secure Sockets Layer (SSL) provides data encryption, server authentication, message integrity, and optional client authentication for a CP/IP connection. You can think of a public key certificate as the digital equivalent of a passport. It is issued by a

trusted organisation, which is known as a certificate authority (CA), and provides identification for the bearer. If, you specify client-certificate authentication, the Webserver will authenticate the client using the client's X.509 certificate, a public key certificate that conforms to
a standard that is defined by X.509 Public Key Infrastructure (PKI). Prior to running an

application that uses SSL, you must configure SSL support on the server and set up the public key certificate.

Question 4:

(i) What do you mean by XML parsing? Briefly describe the parser involved with XML. **(5 marks)**

Answer:

An XML parser (or XML processor) is the software that determines the content and structure of an XML document by combining XML document and DTD (if any present). *Figure below* shows a simple relationship between XML documents, DTDs, parsers and applications. XML parser is the software that reads XML files and makes the information from those files available to applications and other programming languages. The XML parser is responsible for testing whether a document is well-formed and, if, given a DTD or XML schema, whether will also check for validity (i.e., it determines if the document follows the rules of the DTD or schema). Although, there are many XML parsers we shall discuss only Microsoft's parser used by the Internet explorer and W3C's parser that AMAYA uses.

XML Document and their Corresponding DTDs are Parsed and sent to Application

(ii) Compare and contrast SSL and TLS. **(5 marks)**

Answer:

Secure Socket Layer (SSL)/Transport Layer Security (TLS):

Secure Socket Layer (SSL) and Transport Layer Security (TLS), its successor, are cryptographic protocols which provide secure communication on the Internet for as e-mail, internet faxing, and other data transfers.

SSL provides endpoint authentication and communication privacy over the Internet using cryptography. In typical use, only the server is authenticated (i.e. its identity is ensured) while the client remains unauthenticated; mutual authentication requires public key infrastructure (PKI) deployment to clients. The protocols allow client/server applications to communicate in a way designed to prevent eavesdropping, tampering, and message forgery.

In cryptography, message forgery is the sending of a message to deceive the recipient of whom the real sender is. A common example is sending a spam e-mail from an address belonging to someone else.